Subject: clangd
Posted by mirek on Thu, 16 Sep 2021 06:49:19 GMT
View Forum Message <> Reply to Message

Moving the discussion from github. This was said:

Quote:
I use clangd (actually, ccls language server) with U++ for several years.
I use it from vim. This works very well for me.
Opening of a project takes about 10-15 sec on a 8 core (16 threads) CPU.
Navigation is much better than in TheIDE.
Language server spends ~1-2 sec. recompiling a file on each save operation.
I'd recommend to take a look at tree-sitter, which is an incremental parsing library.
Both, language server and tree-sitter are built into a new version of vim called neovim, which I recommend to use as a reference example.
We can also move this discussion to a U++ forum.

WRT speed, I am more concerned about what happens when I change some Core header...

Subject: Re: clangd
Posted by Novo on Thu, 16 Sep 2021 16:42:11 GMT
View Forum Message <> Reply to Message

When I open a project and Core was changed, it takes ~70 sec. to reindex a project (~16 min. of total CPU time).
When there is nothing to reindex, then opening (checking and loading of indexes) a project takes ~10 sec.

Subject: Re: clangd
Posted by Novo on Thu, 16 Sep 2021 17:02:30 GMT
View Forum Message <> Reply to Message

Links:
ccls - C/C++/ObjC language server supporting cross references, hierarchies, completion and semantic highlighting
Tree-sitter is a parser generator tool and an incremental parsing library. It can build a concrete syntax tree for a source file and efficiently update the syntax tree as the source file is edited.
NeoVim.
NeoVim plugins:
Navigator - Navigate codes like a breeze. Exploring LSP and Tree-sitter symbols a piece of cake.
completion-nvim - A async completion framework aims to provide completion to neovim's built in LSP written in Lua.

Subject: Re: clangd
Posted by mirek on Thu, 16 Sep 2021 21:43:05 GMT
View Forum Message <> Reply to Message

Novo wrote on Thu, 16 September 2021 18:42When I open a project and Core was changed, it takes ~70 sec. to reindex a project (~16 min. of total CPU time).
When there is nothing to reindex, then opening (checking and loading of indexes) a project takes ~10 sec.

How big is project? (Is CtrlLib in?)

Mirek

Subject: Re: clangd
Posted by Novo on Thu, 16 Sep 2021 22:06:19 GMT
View Forum Message <> Reply to Message

mirek wrote on Thu, 16 September 2021 17:43Novo wrote on Thu, 16 September 2021 18:42When I open a project and Core was changed, it takes ~70 sec. to reindex a project (~16 min. of total CPU time).
When there is nothing to reindex, then opening (checking and loading of indexes) a project takes ~10 sec.

How big is project? (Is CtrlLib in?)

Mirek
I'd say it is tiny
It took me a couple of hours to make it.
And yes, CtrlLib is included.
I've been talking about this project.

Subject: Re: clangd
Posted by Novo on Thu, 16 Sep 2021 22:19:19 GMT
View Forum Message <> Reply to Message

I checked performance on a non-gui app.
Completely reindex: 5 min. of total CPU time.
Load an already indexed project: 1:30 min. of total CPU time.
This is acceptable on a 8-core (16 threads) cpu.
Basically, this is speed of compilation.

Subject: Re: clangd
Posted by Novo on Thu, 16 Sep 2021 22:20:59 GMT
View Forum Message <> Reply to Message

Well, this non-gui app uses a lot of complicated templates.
So, compilation speed can be slow.

Subject: Re: clangd
Posted by Novo on Thu, 16 Sep 2021 22:25:33 GMT
View Forum Message <> Reply to Message

The project itself opens up immediately. Language Server loads info in background asynchronously.
Navigation is disabled till LS finishes its job.

Subject: Re: clangd
Posted by Novo on Fri, 17 Sep 2021 14:15:52 GMT
View Forum Message <> Reply to Message

You can check performance by yourself.
clangd:
--compile-commands-dir=<string> - Specify a path to look for compile_commands.json. If path is invalid, clangd will look in the current directory and parent paths of each source file

ccls:
--index=<root> - standalone mode: index a project and exit

Do not forget to run "umk -j" to create a compile_commands.json file.

Subject: Re: clangd
Posted by mirek on Fri, 17 Sep 2021 16:49:43 GMT
View Forum Message <> Reply to Message

Novo wrote on Fri, 17 September 2021 16:15You can check performance by yourself.
clangd:
--compile-commands-dir=<string> - Specify a path to look for compile_commands.json. If path is invalid, clangd will look in the current directory and parent paths of each source file

ccls:
--index=<root> - standalone mode: index a project and exit

Do not forget to run "umk -j" to create a compile_commands.json file.

Could you do that for theide?

BTW, that could mean that BLITZ could still work in that context, right?

Mirek

Subject: Re: clangd
Posted by Novo on Fri, 17 Sep 2021 19:09:44 GMT
View Forum Message <> Reply to Message

mirek wrote on Fri, 17 September 2021 12:49
Could you do that for theide?

I do not really understand what you mean by that.
Quote:
BTW, that could mean that BLITZ could still work in that context, right?

Mirek
Correct. Everything that can be compiled with Clang can work in this context.
Just instead of real compilation you need to log Clang commands into a compile_commands.json file.

Current implementation of a "-j" option is not using BLITZ. Most likely, I did that because I couldn't figure out how to do that in umk, and after initial version begun to work I switched to another "let's make it work" project.

Subject: Re: clangd
Posted by mirek on Fri, 17 Sep 2021 20:52:03 GMT
View Forum Message <> Reply to Message

Novo wrote on Fri, 17 September 2021 21:09mirek wrote on Fri, 17 September 2021 12:49
Could you do that for theide?

I do not really understand what you mean by that.

Can you test how long it takes to process theide project (complete, like after Core change)?

Mirek

Subject: Re: clangd
Posted by Novo on Fri, 17 Sep 2021 21:55:27 GMT
View Forum Message <> Reply to Message

mirek wrote on Fri, 17 September 2021 16:52Novo wrote on Fri, 17 September 2021 21:09mirek wrote on Fri, 17 September 2021 12:49
Could you do that for theide?

I do not really understand what you mean by that.

Can you test how long it takes to process theide project (complete, like after Core change)?

Mirek
I'll try. Time should be similar to compilation in Debug configuration (optimization is disabled).

Subject: Re: clangd
Posted by Novo on Fri, 17 Sep 2021 22:17:43 GMT
View Forum Message <> Reply to Message

ccls: 34:30 of total CPU time. Not very fast
Memory usage up to 2.7 GB.
It is a heavy tool.

Subject: Re: clangd
Posted by Novo on Fri, 17 Sep 2021 22:20:05 GMT
View Forum Message <> Reply to Message

Well, parallelization was almost 100%.

Subject: Re: clangd
Posted by mirek on Fri, 17 Sep 2021 22:31:11 GMT
View Forum Message <> Reply to Message

Novo wrote on Sat, 18 September 2021 00:17ccls: 34:30 of total CPU time. Not very fast
Memory usage up to 2.7 GB.
It is a heavy tool.

Needs BLITZ...

Subject: Re: clangd
Posted by Novo on Fri, 17 Sep 2021 22:43:00 GMT
View Forum Message <> Reply to Message

Clang-based indexers are good when you do not have anything else, but they are very slow and memory-hungry.
Theoretically, they should have an ideal quality, but I often have problems with them, starting with inability to compile code from time to time (actually, quite often).
In case, when Clang fails, I just use regular search in project's folders.
But when it works, it is pretty good
A trick is not to save a file very often to prevent reindexing ...

Subject: Re: clangd
Posted by Novo on Sat, 18 Sep 2021 11:46:13 GMT
View Forum Message <> Reply to Message

mirek wrote on Fri, 17 September 2021 18:31
Needs BLITZ...
A catch: you have to add all used h-files to compile_commands.json explicitly.

Subject: Re: clangd
Posted by mirek on Thu, 30 Sep 2021 06:18:55 GMT
View Forum Message <> Reply to Message

Novo wrote on Sat, 18 September 2021 13:46mirek wrote on Fri, 17 September 2021 18:31
Needs BLITZ...
A catch: you have to add all used h-files to compile_commands.json explicitly.

If you send me compile_commands with h-files, I can fix that I think (I mean, add them during compile_commands generation).

Mirek

Subject: Re: clangd
Posted by mirek on Thu, 30 Sep 2021 10:57:03 GMT
View Forum Message <> Reply to Message

After thinking about it for a while, I decided that I am not going to add clangd to TheIDE _myself_:

- it is quite a lot of work
- it does not meet my requirements for speed
- most importantly, I believe that people that want this would actually be even happier with Visual Studio Code (or some other "normal" IDE/editor). To integrate U++ with it seems to be much less work that to add clangd to TheIDE.
- U++ semiheuristic parser is work in progress, but it is MUCH faster than "real compiler" and the current architecture is fine in regards of what it can achieve. I believe that adding missing features to it is in the end less work that to integrate clangd. And much more fun too

That said, if somebody wants to have a try at this, I will be more that happy to watch the progress!

Mirek

Subject: Re: clangd
Posted by Novo on Mon, 04 Oct 2021 19:58:26 GMT
View Forum Message <> Reply to Message

mirek wrote on Thu, 30 September 2021 02:18Novo wrote on Sat, 18 September 2021 13:46mirek wrote on Fri, 17 September 2021 18:31
Needs BLITZ...
A catch: you have to add all used h-files to compile_commands.json explicitly.

If you send me compile_commands with h-files, I can fix that I think (I mean, add them during compile_commands generation).

Mirek
You can easily generate them by yourself. For example for umk:
umk uppsrc umk CLANG -j

In case this doesn't work for you I've attached one generated on my machine.

File Attachments

1) compile_commands.json.gz, downloaded 327 times

Subject: Re: clangd
Posted by mirek on Mon, 04 Oct 2021 21:18:08 GMT
View Forum Message <> Reply to Message

Novo wrote on Mon, 04 October 2021 21:58mirek wrote on Thu, 30 September 2021 02:18Novo wrote on Sat, 18 September 2021 13:46mirek wrote on Fri, 17 September 2021 18:31
Needs BLITZ...
A catch: you have to add all used h-files to compile_commands.json explicitly.

If you send me compile_commands with h-files, I can fix that I think (I mean, add them during compile_commands generation).

Mirek
You can easily generate them by yourself. For example for umk:
umk uppsrc umk CLANG -j

In case this doesn't work for you I've attached one generated on my machine.

Then I do not understand "have to add explicitely"...

Mirek

Subject: Re: clangd
Posted by Novo on Mon, 04 Oct 2021 21:39:18 GMT
View Forum Message <> Reply to Message

mirek wrote on Mon, 04 October 2021 17:18Novo wrote on Mon, 04 October 2021 21:58mirek wrote on Thu, 30 September 2021 02:18Novo wrote on Sat, 18 September 2021 13:46mirek

wrote on Fri, 17 September 2021 18:31
Needs BLITZ...
A catch: you have to add all used h-files to compile_commands.json explicitly.

If you send me compile_commands with h-files, I can fix that I think (I mean, add them during compile_commands generation).

Mirek
You can easily generate them by yourself. For example for umk:
umk uppsrc umk CLANG -j

In case this doesn't work for you I've attached one generated on my machine.

Then I do not understand "have to add explicitely"...

Mirek
When you compile a cpp-project, you compile only cpp/c files. h-files are compiled implicitly via #include directive.
In case of compile_commands.json you need to add them to compile_commands explicitly.
Basically, you need to "compile" h-files as well.

Subject: Re: clangd
Posted by mirek on Mon, 04 Oct 2021 22:18:49 GMT
View Forum Message <> Reply to Message

Novo wrote on Mon, 04 October 2021 23:39mirek wrote on Mon, 04 October 2021 17:18Novo wrote on Mon, 04 October 2021 21:58mirek wrote on Thu, 30 September 2021 02:18Novo wrote on Sat, 18 September 2021 13:46mirek wrote on Fri, 17 September 2021 18:31
Needs BLITZ...
A catch: you have to add all used h-files to compile_commands.json explicitly.

If you send me compile_commands with h-files, I can fix that I think (I mean, add them during compile_commands generation).

Mirek
You can easily generate them by yourself. For example for umk:
umk uppsrc umk CLANG -j

In case this doesn't work for you I've attached one generated on my machine.

Then I do not understand "have to add explicitely"...

Mirek
When you compile a cpp-project, you compile only cpp/c files. h-files are compiled implicitly via #include directive.
In case of compile_commands.json you need to add them to compile_commands explicitly.
Basically, you need to "compile" h-files as well.

So I cannot generate them by umk only, right? I was wondering what you needed to add to make it work...

Mirek

Subject: Re: clangd
Posted by Novo on Tue, 05 Oct 2021 04:24:16 GMT
View Forum Message <> Reply to Message

Sorry, I do not get your question.
My version of compile_commands.json works. It is not optimal because it doesn't use BLITZ. Otherwise it is fine. Everything works.

Subject: Re: clangd
Posted by mirek on Tue, 05 Oct 2021 07:09:33 GMT
View Forum Message <> Reply to Message

Novo wrote on Tue, 05 October 2021 06:24Sorry, I do not get your question.

Is the file you have sent me exactly the same as produced by umk or did you need to edit it to add header files to it?

Mirek

Subject: Re: clangd
Posted by Novo on Tue, 05 Oct 2021 15:21:11 GMT
View Forum Message <> Reply to Message

mirek wrote on Tue, 05 October 2021 03:09Novo wrote on Tue, 05 October 2021 06:24Sorry, I do not get your question.

Is the file you have sent me exactly the same as produced by umk or did you need to edit it to add header files to it?

Mirek
Attached file is 100% automatically generated. I didn't alter it.

Subject: Re: clangd
Posted by mirek on Tue, 05 Oct 2021 15:54:43 GMT
View Forum Message <> Reply to Message

Novo wrote on Tue, 05 October 2021 17:21mirek wrote on Tue, 05 October 2021 03:09Novo wrote on Tue, 05 October 2021 06:24Sorry, I do not get your question.

Is the file you have sent me exactly the same as produced by umk or did you need to edit it to add header files to it?

Mirek
Attached file is 100% automatically generated. I didn't alter it.

it looks like some kind of really weird misunderstanding...

Subject: Re: clangd
Posted by mirek on Mon, 16 May 2022 08:38:53 GMT
View Forum Message <> Reply to Message

mirek wrote on Thu, 30 September 2021 12:57After thinking about it for a while, I decided that I am not going to add clangd to TheIDE _myself_:

- it is quite a lot of work
- it does not meet my requirements for speed
- most importantly, I believe that people that want this would actually be even happier with Visual Studio Code (or some other "normal" IDE/editor). To integrate U++ with it seems to be much less work that to add clangd to TheIDE.
- U++ semiheuristic parser is work in progress, but it is MUCH faster than "real compiler" and the current architecture is fine in regards of what it can achieve. I believe that adding missing features to it is in the end less work that to integrate clangd. And much more fun too

That said, if somebody wants to have a try at this, I will be more that happy to watch the progress!

Mirek

OK, I am reconsidering and willing to give clang a try. However, not clangd - clang ast seems pretty reasonable and I think I can do better using blitz and stuff for performance.

That said, if I run clang for anything nontrivial with dump-ast (that dumps ast in human readable form), it takes 20s to generate 400MB file. Meanwhile, emit-ast is fast and creates 40MB of binary file. Has anybody any experience in parsing this binary data or willing to give it a try?

The ideal solution should at max use what is already in win32 release (- I mean, it is ok to use any .dll from clang). I believe libclang-cpp.dll in fact contains whole clang api, so that should be doable.

Mirek

Subject: Re: clangd

Posted by shawnx on Sun, 27 Nov 2022 17:48:05 GMT
View Forum Message <> Reply to Message

would be great if I can do u++ under vscode somehow, or with vim+clangd, are there some tutorials?

I use bear to generate compile_commands.json then launch vim, seems both clangd and vim are working.

my questions, how can I convert upp to the standard Makefiles? I prefer a standard approach to umks32, do I need convert upp files to Makefile? how should I do that.

Subject: Re: clangd
Posted by Novo on Sun, 27 Nov 2022 21:30:38 GMT
View Forum Message <> Reply to Message

shawnx wrote on Sun, 27 November 2022 12:48I use bear to generate compile_commands.json then launch vim, seems both clangd and vim are working.

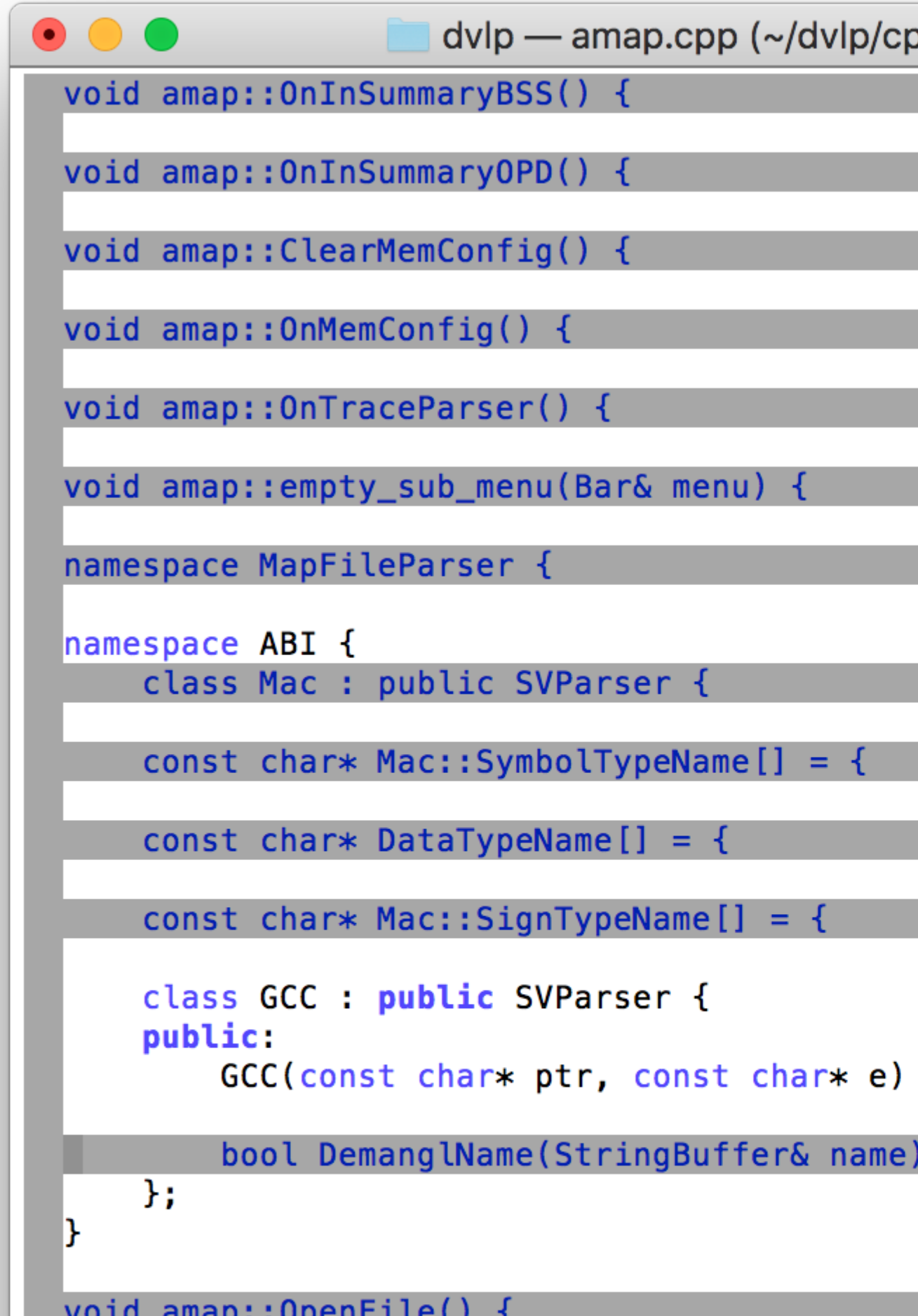
There is no need to use bear with U++. umk has an option "-j" to do that.
Example: "umk MyApps {dirname} CLANG -j"

shawnx wrote on Sun, 27 November 2022 12:48would be great if I can do u++ under vscode somehow, or with vim+clangd, are there some tutorials?

vim+clangd/ccls.

Redefine "make" as "umk MyApps {dirname} CLANG -bsu", so regular ":make" will use umk.
nnoremap <silent> <F7> :make<CR>
You could take a look at tpope/vim-projectionist which allows moving of all configuration settings out of vimrc and creation of per project configurations.
ccls seems to work better than clangd.
Attached image shows my vim session.

File Attachments
1) Screen Shot 2022-11-27 at 4.24.40 PM.png, downloaded 691 times


dvlp — amap.cpp (~/dvlp/cp
void amap::OnInSummaryBSS() {
void amap::OnInSummaryOPD() {
void amap::ClearMemConfig() {
void amap::OnMemConfig() {
void amap::OnTraceParser() {
void amap::empty_sub_menu(Bar& menu) {
namespace MapFileParser {
namespace ABI {
class Mac : public SVParser {
const char* Mac::SymbolTypeName[] = {
const char* DataTypeName[] = {
const char* Mac::SignTypeName[] = {
class GCC : public SVParser {
public:
GCC(const char* ptr, const char* e)
bool DemanglName(StringBuffer& name)
};
}
void amap::OpenFile() {

Subject: Re: clangd
Posted by shawnx on Sun, 27 Nov 2022 21:43:44 GMT
View Forum Message <> Reply to Message

very useful, trying now. I'm excited to learn U++ today.

I have been searching high and low for c++ GUI(Qt, wxwidgets,sciter), somehow U++ was never on the radar, it deserves way more attention. Please brand it as a "BSD version alternative to Qt", it will sell broadly.

Subject: Re: clangd
Posted by Novo on Sun, 27 Nov 2022 21:59:42 GMT
View Forum Message <> Reply to Message

I'm glad you like it.
Below is a list of vim plugins I use with language server.

```
Pack 'm-pilia/vim-ccls', {'type': 'opt'}
" LSC
Pack 'natebosch/vim-lsc', {'type': 'opt'}
" LSP
Pack 'prabirshrestha/asyncomplete.vim', {'type': 'opt'}
Pack 'prabirshrestha/vim-lsp', {'type': 'opt'}
Pack 'jackguo380/vim-lsp-cxx-highlight', {'type': 'opt'}
```

I use LSC at this time although I have configuration settings for both of them.
Interesting reading: LSP in Vim with the LSC Plugin

Subject: Re: clangd
Posted by mirek on Mon, 28 Nov 2022 15:35:50 GMT
View Forum Message <> Reply to Message

shawnx wrote on Sun, 27 November 2022 18:48would be great if I can do u++ under vscode somehow, or with vim+clangd, are there some tutorials?

I use bear to generate compile_commands.json then launch vim, seems both clangd and vim are working.

my questions, how can I convert upp to the standard Makefiles? I prefer a standard approach to umks32, do I need convert upp files to Makefile? how should I do that.

You can also work from vim with umk directly. Use theide as editor just for layouts / images...

That said, if clangd quality of C++ analysis is what you are missing in theide, we are just closing half-year development cycle that replaces our home-grown parser with libclang.

Subject: Re: clangd
Posted by shawnx on Mon, 28 Nov 2022 15:47:51 GMT
View Forum Message <> Reply to Message

That sounds cool, when will it be available?
libclang is not c++20 complete though, for c++17 it shall work really well.

Subject: Re: clangd
Posted by Klugier on Mon, 28 Nov 2022 15:54:15 GMT
View Forum Message <> Reply to Message

Hello shawnx,

We plan to release next version of TheIDE in near future. I can not exactly tell when it will be, but you can test new assist by downloading nightly builds from this site. Just select the newest possible version of your platform of choice.

It works well with C++17 and C++20 is also supported. You can specify version of the parser in Settings -> Assist menu in "libclang additional compiler options" by adding "-std=x" parameter.

Klugier

Subject: Re: clangd
Posted by mirek on Mon, 28 Nov 2022 16:48:39 GMT
View Forum Message <> Reply to Message

shawnx wrote on Sun, 27 November 2022 22:43Please brand it as a "BSD version alternative to Qt", it will sell broadly.

Except it is not exactly. It is completely different approach to the problem....

Qt is for people that have to do GUI in C++.

U++ is for fools that want to do GUI in C++.

Subject: Re: clangd
Posted by mirek on Mon, 28 Nov 2022 16:53:32 GMT

shawnx wrote on Mon, 28 November 2022 16:47That sounds cool, when will it be available?

The plan is to enter "beta" stage at 2022/1/1 and hopefully call it a release by the end of year.

Usually the process is "take nightly build and call it rc[n]". When serious bugs are reported, fix and call it rc[n + 1]. Once nobody complains about it, delete rc bit from the name

Current nightly is good enough for serious development work.

Mirek

Subject: Re: clangd
Posted by Novo on Tue, 13 Dec 2022 00:26:05 GMT

shawnx wrote on Sun, 27 November 2022 12:48would be great if I can do u++ under vscode somehow, or with vim+clangd, are there some tutorials?

I forgot to mention that you need to create a file ~/.vim/compiler/umk.vim which should contain text below.
source $VIMRUNTIME/compiler/gcc.vim
